PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re U.S. Patent Application of:	)	**<u>Group Art Unit</u>: 2137**
	)	
Harald VATER *et al.*	)	**<u>Examiner</u>: Z. Davis**
	)	
Serial Number: 09/700,656	)	***Attorney Docket*: VATE3001/BEU**
	)	
Filed: February 14, 2001	)	**<u>Confirmation No.</u>: 2137**

For: Access-Controlled Data Storage Medium

APPELLANT'S BRIEF UNDER 37 C.F.R. §1.192

Sir:

This paper is an Appeal Brief in furtherance of the Notice of Appeal filed in this case on February 2, 2006.

A petition for a one-month extension of time together with the appropriate fee accompanies this Appeal Brief so that it is timely filed. In addition, the fee required under 37 C.F.R. §1.17(f) is submitted herewith.

This Brief contains these items under the following headings and in the order set forth below:

I. Real Party In Interest
II. Related Appeals And Interferences
III. Status of Claims
IV. Status of Amendments
V. Summary of Claimed Subject Matter
VI. Grounds of Rejection to be Reviewed
VII. Argument
VIII. Claims Appendix
IX. Evidence Appendix

X. Related Proceedings Appendix

I. Real Party In Interest

The real party in interest is Giesecke & Devrient, GmbH, of Munich, Germany.

II. Related Appeals And Interferences

There are no related appeals or interferences.

III.. Status of Claims

The status of the claims in this application is:

A. Status of all the claims

1. Claims canceled: None
2. Claims withdrawn from consideration: 1-25, 34-41, and 43
3. Claims pending: 1-43
4. Claims allowed: None
5. Claims objected to: None
6. Claims rejected: 26-33 and 42

B. Claims on Appeal:

The claims on appeal are: 26-33 and 42

IV. Status of Amendments

No amendments have been submitted subsequent to the rejection mailed July 11, 2008.

V. Summary of Claimed Subject Matter

The claimed subject matter is a method for protecting secret input data in which falsification of the input data (by combination with auxiliary data (Z)) is compensated for by combining output data that results from execution of operations *f* with an auxiliary function value *f*(Z) that has been "*previously determined by execution of the one or more operations a (f) with the auxiliary data (Z) as input data in safe surroundings and stored along with the auxiliary data (Z).* In other words, the claimed invention is to falsify input data and yet achieve the same result as if the input data had not been falsified by using a stored auxiliary function value that was previously generated in safe surroundings. This feature of the invention, recited as the third step of **independent claim 26**, is described in **lines 17-19 on page 3 and lines 6-9 on page 7** of the original specification. The first two steps recited in claim 26 are the falsification of input data and combination of the output with the auxiliary function value, as described for example in **lines 3-17 on page 3 and lines 2-13 on page 7** of the original specification, and illustrated as **steps 9 and 10 of Fig. 3**.

It is of course necessary, whenever falsifying input data, to apply some sort of auxiliary function in order to obtain an intended result, *i.e.*, one that is not affected by the input data falsification. For example, the Kocher publication cited by the Examiner and discussed below (U.S. Patent Publication No. 2002012478), discloses such input data falsification using auxiliary data and auxiliary function values. However, a problem with falsification of input data using auxiliary data and auxiliary function values is that it might be possible for an attacker to determine the auxiliary data and auxiliary function values as they are generated. This problem is solved in the claimed invention by using auxiliary data and auxiliary function values that have been previously calculated in safe surroundings, and stored for use in connection with the input data falsification.

The significance of storing the auxiliary function *f*(Z) in safe surroundings is explained at the end of the second paragraph on page 3 of the present application:

> . . .It is important in this context for the random number and the function value [*i.e.*, the auxiliary data and the auxiliary function value] to be previously determined and stored in safe surroundings so that the calculation of the function value from the random number cannot be intercepted . . .."

Furthermore, as explained in line 7 on page 7 of the original specification, this previous determination could be during production of the card or, more precisely (as explained in the last line of the third paragraph on page 8), during the personalization phase of card production.

As indicated above, **claim 26** positively recites three steps:

- falsifying input data by combination with auxiliary data before the execution of one or more operations;
- combining the output data determined by execution of the one or more operations with an auxiliary function value in order to compensate for the falsification of the input data; and
- the auxiliary function value having been **previously** determined by the execution of the one or more operations with the auxiliary data as input data in **safe surroundings** and **stored** along with the auxiliary data.

The first two steps are disclosed in the prior art, so the claimed invention is the ***combination*** of the first two steps with the third step, which is characterized in that the auxiliary function value is ***previously*** determined by execution of the one or more operations with the auxiliary data as input data ***in safe surroundings***, and by **storage** of the auxiliary function value and auxiliary data, as claimed. The three steps listed above constitute all of the positive limitations in claim 26.

Dependent **claim 27** recites the feature in which falsification of the input is performed before execution of a nonlinear operation g. This feature is described in **lines 11-14 on page 3**, **lines 29-31 on page 6**, **lines 21-23 on page** 7, and especially **lines 1-3 on page 8** of the original specification, and simply refers to the fact that compensation may not be possible after application of a nonlinear rather than linear function (*f* being a linear function).

Dependent **claim 28** recites the feature of varying the auxiliary data to make discovery more difficult, as described in **lines 6-9 on page 9** of the original specification, while **claim 29** recites the feature of generating auxiliary data and functions by combining other auxiliary data and functions, as described in **lines 9-14 on page 9**.

Claim 30 depends from claim 29 recites the feature in which the auxiliary values and functions are selected randomly as described in **lines 6-14 on page 9**.

Dependent **claim 31** recites the feature described in **lines 9-14 on page 9** in which only some of the auxiliary values and functions are stored and others are generated from the previously determined values and functions without having to apply operation *f* to the auxiliary values.

Finally, with respect to the dependent claims, **claims 31 and 32** respectively recite the features, described for example in **lines 3-17-19 on page 3 and lines 6-15 on page 9**, that the auxiliary data are random numbers and the output data are EXORed with the auxiliary function, while **claim 42** recites the feature in which the operations to be protected are key permutations or permutations of other secret data as described in **lines 15-26 on page 8**.

Since the dependent claims all depend from claim 26 and therefore incorporate, by virtue of their dependency, the limitation of claim 26 that the auxiliary function is previously determined in safe surroundings and then stored, all of the claims involved in this appeal include this limitation..

VI. Grounds of Rejection to be Reviewed on Appeal

The rejection to be reviewed on appeal is a rejection of the subject matter of claims 26-33 and 42 as anticipated under 35 USC §103(a) in view of U.S. Patent Publication No. 2001/0053220 (the Kocher publication).

VII. Argument

Reversal of the rejection under 35 USC §103(a) is respectfully requested on the grounds that the Kocher publication and the Vanstone patent, whether considered individually or in any reasonable combination, fail to disclose or suggest the claimed combination of:

- falsifying secret input data by combination with auxiliary data before the execution of one or more operations;
- combining the output data determined by execution of the one or more operations with an auxiliary function value in order to compensate for the falsification of the input data; and
- the auxiliary function value having been previously determined by the execution of the one or more operations with the auxiliary data as input data in safe surroundings and stored along with the auxiliary data,

as recited in claim 26. Claims 27-33 and 42 stand or fall with claim 26.

The Kocher publication discloses the first two steps, but not in combination with **pre-determination** in **safe surroundings** and **storage** of the auxiliary function value and auxiliary data, as claimed. Instead, Kocher discloses that the auxiliary data and auxiliary function value are computed **while** computing the permutation of the input data. As a result, the auxiliary function value and data are vulnerable to inspection by an attacker. The Vanstone patent does not make up for the failure of the Kocher publication to disclose implementation of the first two steps of the claimed method using a **previously determined** auxiliary function value, since the Vanstone patent does not disclose any sort of input data falsification with auxiliary data or compensating auxiliary function value, and therefore could not have suggested that the auxiliary function value should be previously determined and stored along with the auxiliary data, as claimed.

In reply, the Examiner argues in the first paragraph on page 3 of the final Office Action that a combination of the steps disclosed in Kocher (namely the first two steps recited in claim 26) with the "general teaching in Vanstone et al. of pre-computation of secret values" would render the present invention unpatentable. In addition, the Examiner argues in the paragraph bridging pages 3 and 4 of the Official Action, that while Vanstone would admittedly not expressly disclose the feature or pre-computing the auxiliary functions as claimed, the "*broad teaching within Vanstone et al. Of pre-computing and storing secret values securely*" would have given the skilled person the suggestion to modify the teaching of Kocher to obtain the claimed invention. The defects in these arguments are that

(1) what is claimed is not just the pre-computation of secret values, but also the pre-computation of auxiliary functions by performing operations (*f*) on the secret values,

(2) Vanstone does not actually teach pre-computation of secret values in general, but rather pre-computation of one particularly computation-intensive value and real time generation of the remaining values, *whether secret or not.*

In other words, Vanstone does not care whether a value is secret or not, but rather teaches pre-computation to save time, based on whether the value is computation-intensive and not on whether it is secret. In fact, the values that are not pre-computed by Vanstone, including the secret key k (!!), are at least as secret as the value that is pre-computed, the decision to pre-compute being based solely on computation intensiveness and not a need for secrecy. If Vanstone's pre-computation of γ was for the purpose of protecting secret values by pre-computation, then Vanstone would also have pre-computed the key k generated by using γ.

Instead of being concerned with protecting secret values, the Vanstone publication is specifically directed to a method of efficiently computing a session parameter for use with public key protocols (see col. 1, lines 12-15 of Vanstone). This is done by multiplicatively decomposing the respective session parameter a^x into two parts, a^x and γ, where k is an integer representing the secret key, a is a generator of a mathematically defined structure called a group, and $\gamma = a^1$, where k + k' + i. The value γ, which is computationally expensive to computer because of the relatively high Hamming weight (i.e., number of 1's in the binary code) of i may be pre-computed and stored, as explained in col. 2, lines 6-22 and col. 3, lines 16-27. In other words, Vanstone specifically teaches that a particularly difficult-to-compute value used in generating a session parameter may be pre-computed to save time during actual session key generation. Vanstone's reason for pre-computing γ has nothing to with protecting auxiliary functions of the type disclosed in Kocher, and is not obviously applicable thereto. In fact, Vanstone is only concerned with computational efficiency and not with whether a value is secret or not.

The Kocher publication, on the other hand, not only fails to teach pre-computation of the auxiliary functions, but to the contrary teaches that auxiliary functions may be generated during falsification of the input data. Thus, while Kocher teaches the first two steps of the claimed method, Kocher teaches that the third step of the claimed method, namely computing the auxiliary data *is carried out* **while** *computing the permutation of the input data.*[1] In other words, in contrast to the

[1] The exact method of blinding used by Kocher is given in the form of source code included in paragraph [0068], as follows:

- Suppose the length of the input array dataIn is 64. An array *perm* holding the numbers from 0 to 63 in a randomly permuted manner is computed in a first step (see the first and second for-loops of the listed source code).
- In a second step, another array *temp* of length 64 is computed by setting for every I from 0 to 63:
  temp [p] := dataIn [p] ^ b,
  where ^ denotes the XOR operator, b is a random bit just computed, and p = perm [I], *i.e*, p is an index given by the previously computed random array perm.
- In a third step (also in the second for-loop), dataOut is temporarily set to dataOut [table [p]] = b (see the third for-loop of the listed source code).
- In a fourth step, the array perm is recomputed, *i.e.*, randomly permuted once more (see the fourth for-loop), although the general algorithm would work without this step.
- Finally, in the last step, the input data dataIn is finally permuted by setting, for every I from 0 to 63, where again p = perm[i]:
  dataOut [table [p]] := temp [p], which is the same as
  dataOut [table [p]) := dataIn[p] ^ b (see the second step above).

The blinding of the input data by the random bit b is compensated for by setting:
dataOut [table [p]] ^:= temp [p], which is a shorthand notation for
dataOut [table[p]] := dataOut [table[p]] ^ temp [p].

Summarizing, one obtains from the temp [p] correlations and the expression dataOut [table [p]] = b obtained in the third step described above:
dataOut [table[p]] := b ^ dataIn[p] ^ b = dataIn[p], which is what was originally to be computed since b ^ b = 0.

It can be seen from this analysis of the blinding method listed in paragraph [0068] of Kocher that only the order in which the operations dataOut [table [p]] := dataIn [p] were performed is permuted by the random permutation *perm*, by setting p = perm [I] where I runs from 0 to 63. Furthermore, the input bits dataIn [p], where temporarily blinded by the random bits b are compensated for after the permutation of the blinded input bits.

In particular, the first step of the claimed method, namely falsifying of input data (dataIN), is carried out by combination with auxiliary data in the form of random bits b before the execution of one or more operations (the permutation of the array dataIN according to the order given by the source code). The second step of the claimed method, namely compensation for the falsification, is carried out by determining the output data by the execution of one or more operations (dataIn [p] ^ b = temp [p]) combined using dataOut [table [b] ^= temp [p] with an auxiliary function value (dataOut [table [b]] = b). However, the third step, namely computing present invention, the auxiliary data and the auxiliary function value are not previous computed in safe surroundings, as claimed, but rather are computed internally and thus may relatively easily be inspected by an attacker upon analysis of signal patterns reflective of the auxiliary data and function value computation.

Even if Kocher's auxiliary values were previously generated, rather than supplied by a random number generator during compensation, there is nothing in Vanstone to suggest pre-computation of the auxiliary functions in addition to the auxiliary values. As explained below, Vanstone merely teaches pre-computation of one computationally intensive value with a high Hamming number, and not pre-computation of all secret values, much less auxiliary functions that use secret auxiliary values, as claimed. In other words, even if Vanstone suggested pre-computation of secret values in general, rather than just values with high Hamming numbers (whether secret or not!), the principle that secret **values** should be pre-computed does not imply that the auxiliary **functions** need to be pre-computed.

In addition to failing to provide any teaching that would have led the ordinary artisan to disregard Kocher's specific teaching of generating auxiliary functions while computing the permutation of input data, the Vanstone publication is from a technical field that is remarkably different from that of Kocher and the claimed invention (contrary to the Examiner's argument in the second paragraph on page 4 that Vanstone is reasonably pertinent to the problems of the present invention because it involves "*prevention of secure values used in encryption and other secure calculations*." As noted above, Vanstone is not concerned with protection of secret values by detecting computations since Vanstone does not bother to protect the secret key k from being spied out by intercepting the computation of $k = k' + i$ (protecting γ without protecting k is like protecting a diamond polishing cloth (γ) while leaving the diamond itself (k) unprotected). Furthermore, the

the auxiliary data (random bits b) and auxiliary function value (the permutation of b temporarily stored as dataOut [table [p]]), *is carried out* while *computing the permutation of the input data.*

technical field of Vanstone involves efficient computation of a "secure session key," *i.e.*, a session key that cannot be broken by a potential attacker who is intercepting some communication channel outside the respective computing environment on which the session key is computed (see col. 1, lines 33-39 of Vanstone), and who can read the respective session key but not break it because of the computational difficulty of the underlying mathematical problem. In contrast, the claimed invention and Kocher are concerned with protecting secret data, by falsifying it by means of auxiliary data while the secret data is processed inside a data carrier, from an attacker who is contacting the internal structures of the data carrier in order to intercept signals produced by the specific hardware. The present invention does not involves insoluble *mathematical* problems, but rather how to protect a particular computing environment from hardware attacks, whereas Vanstone is not concerned with preventing interception of the encrypted data.

On pages 4-5 of the Official Action, the Examiner responds to the Appellant's previous argument concerning Vanstone's use of the term "computationally secure" by stating that Vanstone does mention maintaining secrecy, and that secure is sometimes used in the general sense of protecting values. Irrespective of the specific meaning of the term "secure" in Vanstone, however, it is clear that Vanstone's teaching of pre-computation is not for the purpose of security, but rather is solely for computational efficiency. Of course, Vanstone seeks to maintain data security in general. That is the purpose of encryption. However, Vanstone does not extend this concern to the type of attacks with which the present invention is concerned. This is evident from the lack of any teaching in Vanstone of pre-computing the encryption key k, as opposed to just the value γ, in order to prevent an observer from deducing k by observing the computation i + k' (incidentally, this computation, which the Examiner indicates could not be found in Vanstone, is described in col. 3, line 28 of the Vanstone patent). The reason Vanstone gives for not pre-computing k is not that it does not need to be kept secure, but rather that it is selected to have a low Hamming weight (see col. 2, lines 7-19).

Finally, starting at page 5 of the Official Action, the Examiner argues that the skilled person would have been motivated to combine the teachings of Kocher and Vanstone for the purpose of **accelerating the computation** (assuming that the combination would have resulted in the claimed invention, which is not the case as argued above). This conjecture on the part of the Examiner is not generally correct. Vanstone does not teach how to speed up the computation of an arbitrary value, but only how to speed up computation of a certain value of the form a^k where the integer k has a large Hamming weight. This teaching is not applicable to either the claimed invention or that of Kocher since no value having the form a^k needs to be computed. As a result, Vanstone's specific teachings concerning computation acceleration of are no relevance to, and would not have provided motivation for, the proposed combination. If anything, having to retrieve pre-computed values in the computationally simple method of Kocher might actually slow down computation, since Kocher's computation can be performed by a processor with on-board registers without the need to access a disk or flash memory. From the description of the source code in paragraph [0068] of the Kocher publication, it does not appear that there would be any performance gain from reading the random values temp [i] and dataOut [i] in the first "for"-loop of Kocher and the random bits b in the third "for"-loop of Kocher from a disc where they were pre-computed and stored, rather than generating them using some optimized library function such as trueRandom() as taught by Kocher. Vanstone pre-computes because of the large numbers that must be manipulated to generate γ, whereas there is no reason to believe that Kocher's preferred number generator is slower than retrieving pre-computed values from storage. The purpose of using pre-stored values in the claimed invention is not to provide computational efficiency, but rather to enable both auxiliary values and auxiliary functions to be protected from attacks that seek to discover how the values and functions are generated–which is a problem not addressed by either Kocher or Vanstone, either specifically or generally.

In summary, Kocher teaches falsifying secret input data by combination with auxiliary data before the execution of one or more operations *and* combining the output data determined by execution of the one or more operations with an auxiliary function value in order to compensate for the falsification of the input data; but not the feature in which the auxiliary function value has been previously determined by the execution of the one or more operations with the auxiliary data as input data in safe surroundings and stored along with the auxiliary data. Vanstone teaches that a computationally intensive component of a secret key may be pre-computed and stored, so as to make it easier to generate the key when needed, but teaches nothing that would suggest pre-computing and storing the auxiliary values used by Kocher, which are not pre-computed and stored (as explained in more detail in the previous response). Therefore, it is respectfully submitted that one of ordinary skill in the art would not have found it obvious to apply the teachings of Vanstone to the modify the method of Kocher, as claimed, and even if the respective teachings were to have someone been combined, the claimed invention would not have resulted. Reversal of the rejection of claims 26-33 and 42 is accordingly requested.

Conclusion

For all of the foregoing reasons, Appellants respectfully submit that the Examiner's final rejections of claims 26-33 and 42 under 35 U.S.C. §102(e) are improper and should be reversed by this Honorable Board.

Respectfully submitted,

BACON & THOMAS, PLLC

Date: January 14, 2009 By: BENJAMIN E. URCIA
Registration No. 33,805

BACON & THOMAS
625 Slaters Lane, 4th Floor
Alexandria, Virginia 22314

Telephone: (703) 683-0500

S:\Producer\bcu\Pending Q...Z\V\VATER 700656\ab.wpd

VIII. APPENDIX OF CLAIMS ON APPEAL

26. A method for protecting secret data serving as input data for one or more operations, comprising the steps of:

- falsifying the input data by combination with auxiliary data (*Z*) before execution of one or more operations (*f*),
- combining the output data determined by execution of the one or more operations (*f*) with an auxiliary function value (*f*(*Z*)) in order to compensate for the falsification of the input data,
- wherein the auxiliary function value (*f*(*Z*)) was previously determined by execution of the one or more operations (*f*) with the auxiliary data (*Z*) as input data in safe surroundings and stored along with the auxiliary data (*Z*).

27. A method according to claim 26, wherein the combination with the auxiliary function values (*f*(*Z*)) for compensating the falsification is performed at the latest directly before execution of an operation (*g*) which is nonlinear with respect to the combination generating the falsification.

28. A method according to claim 26, wherein the auxiliary data (*Z*) are varied, the corresponding function values being stored in ~~the~~ a memory of ~~the~~ a data carrier.

29. A method according to claim 28, wherein new auxiliary values (*Z*) and new auxiliary function values (*f*(*Z*)) are generated by combining two or more existing auxiliary data (*Z*) and auxiliary function values (*f*(*Z*)).

30. A method according to claim 29, wherein the two or more existing auxiliary data (*Z*) and auxiliary function values (*f*(*Z*)) that are combined to generate the new auxiliary values (*Z*) and new auxiliary function values (*f*(*Z*)) are each selected randomly.

31. A method according to claim 26, wherein pairs of auxiliary data (*Z*) and auxiliary function values (*f*(*Z*)) are generated by a generator without the operation (*f*(*Z*)) being applied to the auxiliary data (*Z*).

32. A method according to claim 26, wherein the auxiliary data (*Z*) are a random number.

33. A method according to claim 26, wherein the output data and the auxiliary function value are combined by an XOR operation.

42. A method according to claim 26, wherein the security-relevant operations are key permutations or permutations of other secret data.

IX. EVIDENCE APPENDIX

No evidence is submitted herewith.

X. RELATED PROCEEDINGS APPENDIX

No related proceedings have occurred, and none are pending.